2.8 CM/CCM

Countermeasures are means of exploiting an adversary's own activity as a means of determining his intentions and/or reducing his effectiveness. Such means involve using the energy radiated as a result of the adversary's presence or activities to detect and identify his weapons systems and platforms. Also, weaknesses in sensors, fuzes, computer processing, and personnel training can be exploited in order to deceive an adversary as to the actual tactical situation or to cause malfunction of his equipment.

Types of radiating systems include communications systems and sensors. Generally, sensors receive more attention (in terms of CM/CCM) since enemy sensors are primarily directed toward friendly forces whereas enemy communication is directed only at its own side.

Countermeasures against radiating systems generally consist of either noise or deception jamming. Noise and deception jamming are processes by which an enemy sensor or communications device is reduced in effectiveness. Traditionally, there are the “four horsemen” of countermeasures: deny, delay, degrade, and destroy. This discussion focuses on the first three since destruction is included under weapon effects.

Electronic Countermeasures (ECM)

The purpose of ECM is to lower the effectiveness of enemy sensors and communications devices. There are three basic methods by which this is accomplished:

a. Direct energy at a receiver to either cause intermittent effectiveness or to generate false targets and deceive equipment operators.

b. Modify the properties of the medium between a communication or sensor device and its intended target.

c. Modify the reflective signature of the target using active or passive measures. These techniques can often be used in the visual spectrum to change the relative contrast of the target.

Method a is commonly referred to as “jamming.” Jamming is achieved by saturating the target receiver (either sensor or communications) with noise so that no information is gained from it. The major difference between different jamming techniques deal with how the jamming energy is delivered and depends on the desired effect.

Barrage jamming is a technique whereby energy is directed at enemy receivers in as wide a band as possible. Barrage jamming attempts to influence, at least to some minimal extent, all receivers in the vicinity that fall within the spectrum of the jammer. This can cause collateral effects on friendly systems. Spot jamming involves directing energy in a narrow band as close to the receiver’s center frequency as possible. Spot jamming is normally meant to counter a specific receiver when the receiver’s center frequency is known from some other collection mechanism. Spot jamming can be more effective than barrage jamming while using much less power due to the narrowness of the transmission band. Sweep jamming is a combination of the previous techniques where a narrow signal is swept through a wider bandwidth in order to disrupt reception of several bands in an intermittent manner. Sweep jamming is often referred to as slow swept or fast swept, where the relative times are dependent upon the inherent time constants of the systems being attacked with the jamming energy.

Modification of the transmission medium (Method b) is usually accomplished using chaff or flares. Chaff provides spurious EM reflections to radar equipment, providing an effective “screen” for aircraft. Flares are used to confuse IR guidance systems by causing “hot spots” in the IR background, thus distracting IR detectors.

The signature of a target can be modified (Method c) in several ways. Radar-absorbing paint is used to weaken reflections. Irregular physical surface structures cause reflection paths to be (at best) orthogonal to the line of site and (at least) wide enough from the incident path to effectively shrink the signature of the target.

2.8.1 Functional Element Design Requirements

SWEG shall allow the user to define countermeasure systems for electro-magnetic and acoustic sensors and electro-magnetic communications.

There shall be no arbitrary limit on the number of countermeasure systems that can be a part of a player structure other than that imposed by the host computer environment.

SWEG shall provide two levels of detail for countermeasure systems: explicit sensor and communications (for signal to noise plus jamming calculations) and implicit sensor communications (for probabilistic loss of lock by a sensor tracker).

SWEG shall provide a flexibility for explicit countermeasure systems representation that is similar to sensors and communications. For example, asymmetric antenna patterns, frequency dependent gains, and terrain masking effects.

SWEG shall provide the capability to allow for an optional dimension in the kill probability table to account for countermeasure effects versus a weapon that cannot be adequately represented with implicit or explicit sensor countermeasures.

SWEG shall account for relative geometry effects on explicit countermeasure systems interactions with sensors and communications. In the case of sensors the interactions will be on a sensor chance basis. For communications the interaction will be on a message basis.

2.8.2 Functional Element Design Approach

Countermeasures are represented as systems that belong to elements in the player structure. There are no explicit player structure linkages between countermeasure systems and other systems. Thinker systems can employ countermeasure systems based on the defined tactics and times to think. Other than this, there are no other control relationships with countermeasure systems.

Countermeasure systems (called disruptors in SWEG) do not have their own event, such as weapons, sensors, thinkers, and communication systems. Instead, disruptors are checked and included if necessary in sensor and communication events. Disruptors can be turned on and off by the user via SDB instructions in a pre-planned mode, or turned on and off within the simulation using tactics defined by the user. Both modes can be used for the same disruptor system but this is not recommended because of the likelihood of an inconsistency between the pre-planned instructions and the reactions taken by the player.

Pre-planned disruptor use is limited to either turning the disruptor on or off or setting it non-operational. Reactive changes allow for multiple simultaneous spots or frequency bands for a single disruptor. The center frequencies of each spot will be centered on the perceived frequency of an emitter. Although the disruptor will affect the sensor or communication receiver, it is employed by using perceptions of the emitting transmitters.

Disruptors can affect multiple receivers simultaneously, and a disruptor can affect both sensors and communications simultaneously via multiple spots or with a single spot if both the sensor and communication receiver are within the band defined by the spot. Disruptor effects are independent of the defined sides and command chains so collateral effects can occur.

Explicit disruptors can be affected by terrain line of sight using common algorithms with sensors and communications. Similarly, disruptor antenna patterns are defined by common data formats and used with algorithms also used with sensors and communications.

Energy propagation is instantaneous, similar to sensors and communications. Implicit disruptors, and disruptors represented as dimensions in the kill probability table have no energy transmission represented.

Explicit disruptors can be detected by sensors with the proper characteristics.

2.8.3 Functional Element Software Design

This section contains one table and two software code trees which describe the software design necessary to implement CM/CCM elements. Table 2.8-1 lists most of the functions found in the code trees, and a description of each function is provided. Figure 1.5-1 describes the top level C++ functions in the code for CM/CCM elements. It contains the path from main to the main jammer function jamcal. Figure 1.5-2 is the detailed code trees for these functions.

A function's subtree is provided within the figure only the first time that the function is called. Some functions are extensively called throughout SWEG, and the trees for these functions are in the appendix to this document rather than within each FE description. Within this FE, the functions in that category are the member functions in the C++ class WhereIsIt**.**

Not all functions shown in the figures are included in the table. The omitted entries are trivial lookup functions (single assignment statements), list-processing or memory allocation functions, or C++ class functions for construction, etc.

TABLE 2.8-1. CM/CCM Elements Functions Table.

Function	Description
BaseHost::Run	runs all steps
BSRVcalculate	determines sensor result using signal/noise calculations
BSRVevent	controls sensor physical processing
BSRVinitialize	initializes physical sensor processing
BSRVonechance	supervises sensor chance calculations
BSRVsystsensing	performs physical system sensing calculations
jamcal	calculates jammer interference power at victim receiver
main	controls overall execution
MainInit	initiates processing and runs either the boot step or normal execution
MainParse	controls parsing of user instructions
program	controls execution of all steps except bootstrap
semant	controls semantic processing of instructions
simnxt	controls event sequencing and runtime execution
simphy	controls processing of physical events
simul8	controls semantic processing of runtime instructions
TAddrData::GetJamInteractions	checks for jammer interactions with a communications device
TAddrData::GetParentData	retrieves the TAddrData object from a parent
TMemory::Deallocate	deallocates a list of blocks by using the address within the provided pointer
TMemory::DeallocFront	deallocates storage
TTable::SearchInt	searches a table for a specific integer
WhereIsIt::CalcPosition	determines position for a platform given a time
yakker	determines signal level at receiver

```
main
    |-BaseHost::Run
        |-MainInit
            |-program
                |-MainParse
                    |-semant
                        |-simul8
                            |-simnxt
                                |-simphy
                                    |-BSRVevent
                                    |   |-BSRVinitialize
                                    |   |-BSRVonechance
                                    |       |-BSRVcalculate
                                    |           |-jamcal
                                    |           |-BSRVsystsensing
                                    |               |-jamcal
                                    |-yakker
                                        |-jamcal
```

FIGURE 2.8-1. CM/CCM Elements Top Level Code Tree.

```
jamcal
    |-TMemory::Index2Ptr
    |-WhereIsIt::CalcPosition
    |-TAddrData::GetParentData
    |   \-TAddrNode::GetParentData
    |-TAddrData::GetJamInteractions
    |-TTable::SearchInt
    |   \-TMemory::Ptr2Index
    \-TMemory::Deallocate
        |-TMemory::DeallocFront
        |   \-TMemory::GetBlockLength
        |-TMemory::Index2Ptr
        |-CountMemOpns
        |   \-TMaster::DebugOn
        \-TMemory::RcylBlock
            |-TMemory::Index2Ptr
            \-TMemory::Ptr2Index
```

FIGURE 2.8-2. CM/CCM Elements Code Tree.

2.8.4 Assumptions and Limitations

- Energy transmission is instantaneous.
- Energy use is not explicitly represented.
- The speed of light is 299.
- 792,792,800 m/sec and the speed of sound is 330.28 m/sec.
- Root mean square signal power is used in all calculations.
- Received power is uniformly distributed over the receiver bandwidth.

2.8.5 Known Problems or Anomalies

None.